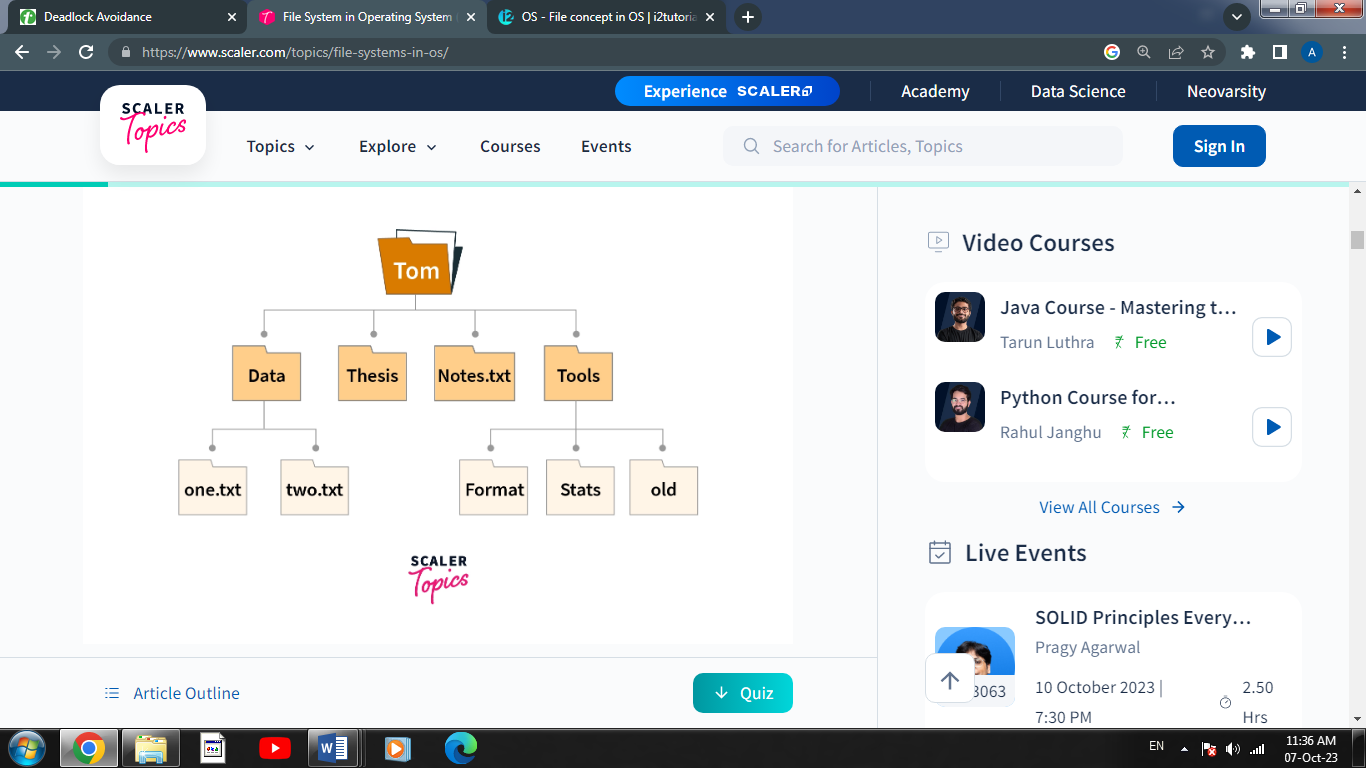
**Unit 5 : File Systems and I/O System**

* **What is a File system in OS?**

A file system in OS dictates how the contents of a storage medium are stored and organized. These storage media (such as secondary memory, external drives, etc) could be computer secondary memory, flash memory, etc. The contents are either files or directories. Most of the time, a storage device has a number of partitions. Each of these partitions is formatted with an empty file system for that device. A file system helps in separating the data on the storage into comparatively smaller and simpler segments. These chunks are files and directories. The file system also provides for storing data related to files, such as their name, extension, permissions, etc.

E.g

* **What is the file?**

The file can be explained as the smallest unit of storage on a computer system. The user can perform file operations like open, close, read, write, and modify.

* **Files-basic Concept**

The operating system can provide a logical view of the information stored in the disks, this logical unit is known as a file. The information stored in files is not lost during power failures.

Files are typically identified by a unique name and extension, which helps users and the operating system understand their content and format. For example, a file named "document.txt" is recognized as a text document by its ".txt" extension.

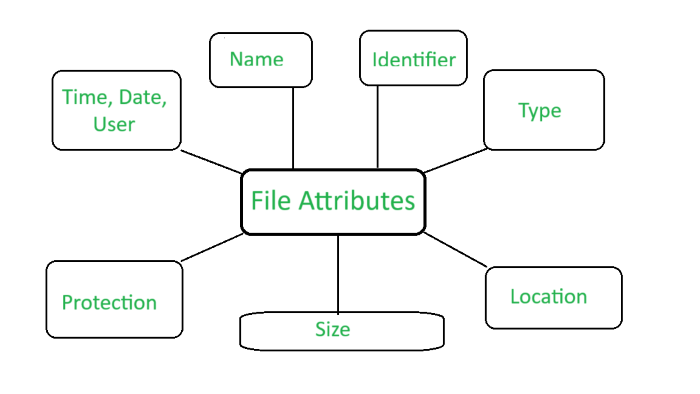
Files can be organized into directories or folders, forming a hierarchical structure that helps users manage and locate their data. This directory structure allows for better organization and easy retrieval of files.

Operating systems provide a set of commands and tools to perform various operations on files, such as creating, opening, editing, moving, copying, and deleting. These operations are crucial for managing data effectively and efficiently.

Furthermore, files often have associated metadata, which includes information like the file's size, creation date, modification date, and permissions. Permissions determine who can access, modify, or delete a file, adding a layer of security to the system.

A file helps to write data on the computer. It is a sequence of bits, bytes, or records, the structure of which is defined by the owner and depends on the type of the file.

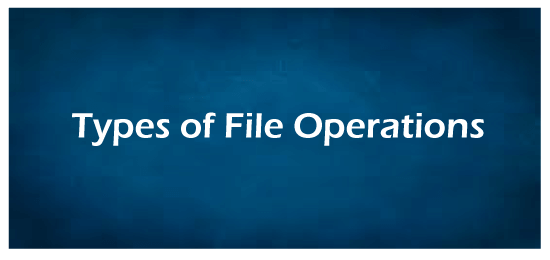
* **File Attributes**

****

1. **Name:** File name is the name given to the file. A name is usually a string of characters.
2. **Identifier**: Identifier is a unique number for a file. It identifies files within the file system. It is not readable to us, unlike file names.
3. **Type:** Type is another attribute of a file which specifies the type of file such as archive file (.zip), source code file (.c, .java), .docx file, .txt file, etc.
4. **Location:** Specifies the location of the file on the device (The directory path). This attribute is a pointer to a device.
5. **Size:**Specifies the current size of the file (in Kb, Mb, Gb, etc.) and possibly the maximum allowed size of the file.
6. **Protection:**Specifies information about Access control (Permissions about Who can read, edit, write, and execute the file.) It provides security to sensitive and private information.
7. **Time, date, and user identification**: This information tells us about the date and time on which the file was created, last modified, created and modified by which user, etc.

* **Operations on the File**

A file is a collection of logically related data that is recorded on the secondary storage in the form of sequence of operations. The content of the files are defined by its creator who is creating the file. The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. These operations are performed by the user by using the commands provided by the operating system. Some common operations are as follows:



**1. Create operation:** This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.

**2. Open operation:** This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

**3. Write operation:** This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.

**4. Read operation:** This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

**5. Re-position or Seek operation:**  The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

**6. Delete operation:** Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.

**7. Truncate operation:** Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.

**8. Close operation:** When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.

**9. Append operation:** This operation adds data to the end of the file.

**10. Rename operation:** This operation is used to rename the existing file.

### ***Files Attributes and Their Operations***

| **Attributes** | **Types** | **Operations** |
| --- | --- | --- |
| Name | Doc | Create |
| Type | Exe | Open |
| Size | Jpg | Read |
| Creation Data | Xis | Write |
| Author | C | Append |
| Last Modified | Java | Truncate |
| protection | class | Delete |

* **Types of files**

The types of files recognized by the system are either **regular**, **directory**, or **special**. However, the operating system uses many variations of these basic types.

***The following basic types of files exist:***

| **Item** | **Description** |
| --- | --- |
| **regular** | Stores data (text, binary, and executable) |
| **directory** | Contains information used to access other files |
| **special** | Defines a FIFO (first-in, first-out) pipe file or a physical device |

All file types recognized by the system fall into one of these categories. However, the operating system uses many variations of these basic types.

1. **Regular files**

Regular files are the most common files and are used to contain data. Regular files are in the form of text files or binary files:

1. ***Text files***

Text files are regular files that contain information stored in ASCII format text and are readable by the user. You can display and print these files. The lines of a text file must not contain NUL characters, and none can exceed {LINE\_MAX} bytes in length, including the newline character.

The term *text file* does not prevent the inclusion of control or other nonprintable characters (other than NUL). Therefore, standard utilities that list text files as inputs or outputs are either able to process the special characters or they explicitly describe their limitations within their individual sections.

1. ***Binary files***

Binary files are regular files that contain information readable by the computer. Binary files might be executable files that instruct the system to accomplish a job. Commands and programs are stored in executable, binary files. Special compiling programs translate ASCII text into binary code.

Text and binary files differ only in that text files have lines of less than {LINE\_MAX} bytes, with no NUL characters, each terminated by a newline character.

1. **Directory files**

Directory files contain information that the system needs to access all types of files, but directory files do not contain the actual file data. As a result, directories occupy less space than a regular file and give the file system structure flexibility and depth. Each directory entry represents either a file or a subdirectory. Each entry contains the name of the file and the file's index node reference number (*i-node number*). The i-node number points to the unique index node assigned to the file. The i-node number describes the location of the data associated with the file. Directories are created and controlled by a separate set of commands.

1. **Special files**

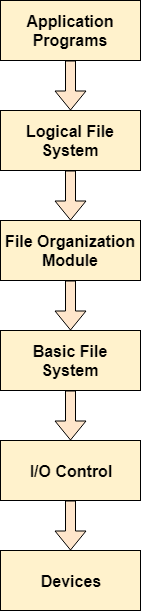
Special files define devices for the system or are temporary files created by processes. The basic types of special files are FIFO (first-in, first-out), block, and character. FIFO files are also called *pipes*. Pipes are created by one process to temporarily allow communication with another process. These files cease to exist when the first process finishes. Block and character files define devices.

* **File System Structure**

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



* When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
* Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.
* Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
* I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.
* **What are File Access Methods in OS?**

A file is a collection of bits/bytes or lines that is stored on secondary storage devices like a hard drive (magnetic disks).

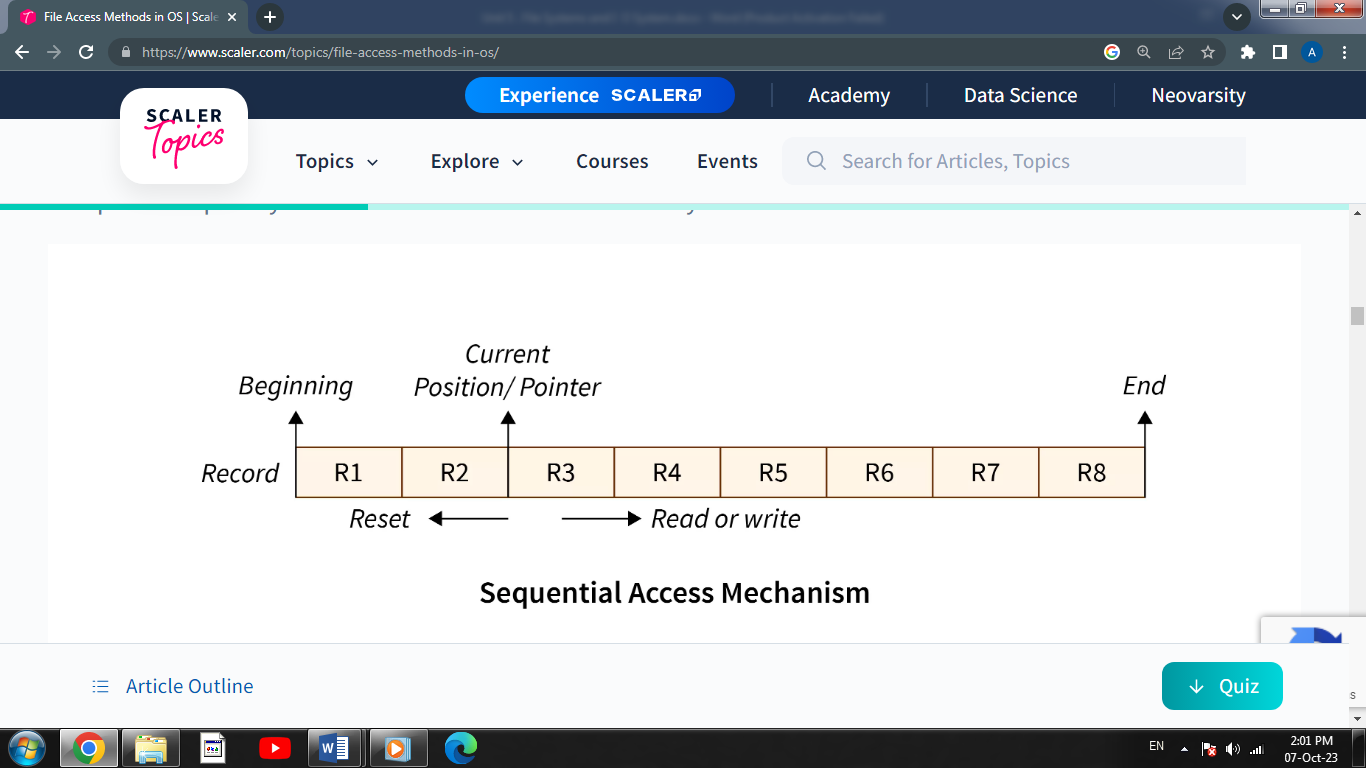
File access methods in OS are nothing but techniques to read data from the system's memory. There are various ways in which we can access the files from the memory like:

* **Sequential Access**
* **Direct/Relative Access, and**
* **Indexed Sequential Access.**

## Types of File Access Methods in the Operating System

### **Sequential Access**

The operating system reads the file word by word in a sequential access method of file accessing. A pointer is made, which first links to the file's base address. If the user wishes to read the first word of the file, the pointer gives it to them and raises its value to the next word. This procedure continues till the file is finished. It is the most basic way of file access. The data in the file is evaluated in the order that it appears in the file and that is why it is easy and simple to access a file's data using a sequential access mechanism. For example, editors and compilers frequently use this method to check the validity of the code.



***Advantages of Sequential Access:***

* The sequential access mechanism is very easy to implement.
* It uses lexicographic order to enable quick access to the next entry.

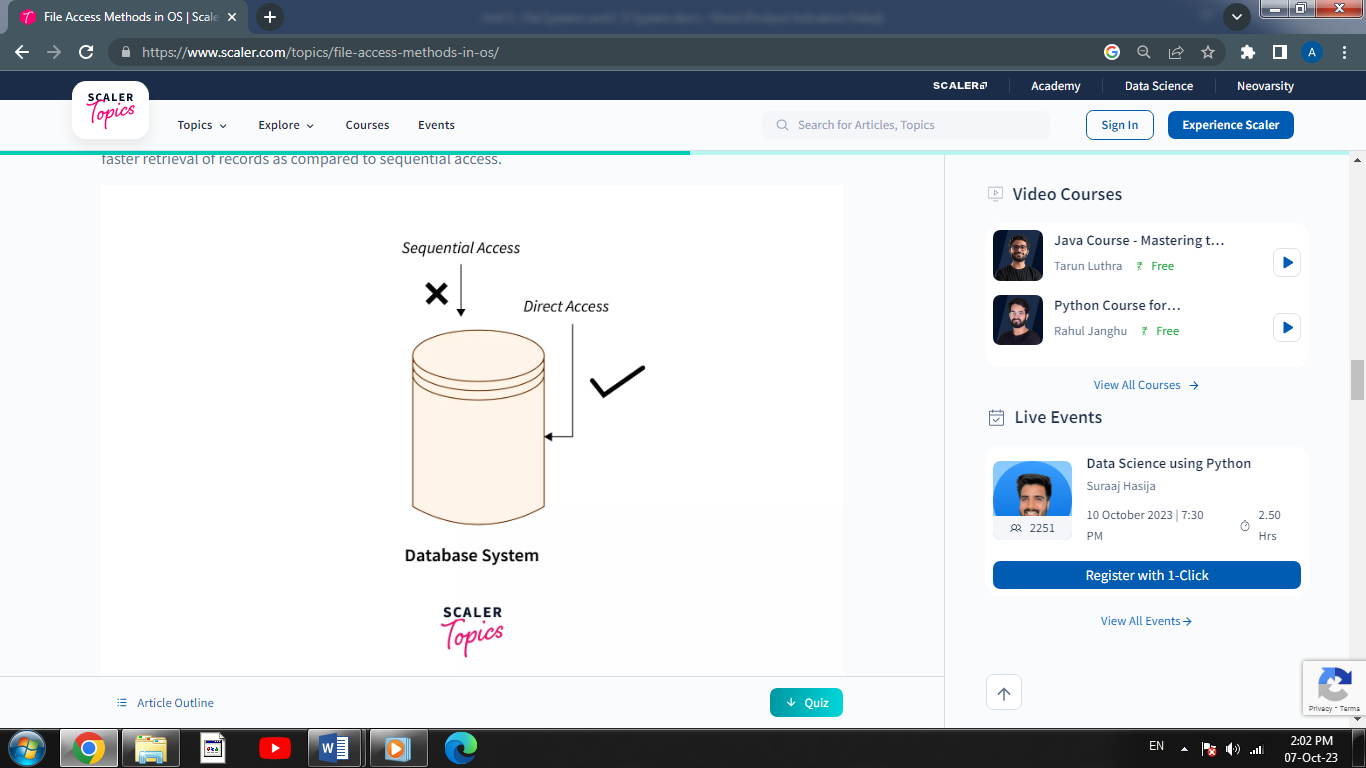
***Disadvantages of Sequential Access:***

* Sequential access will become slow if the next file record to be retrieved is not present next to the currently pointed record.
* Adding a new record may need relocating a significant number of records of the file.

### **Direct (or Relative) Access**

A Direct/Relative file access mechanism is mostly required with the database systems. In the majority of the circumstances, we require filtered/specific data from the database, and in such circumstances, sequential access might be highly inefficient. Assume that each block of storage holds four records and that the record we want to access is stored in the tenth block. In such a situation, sequential access will not be used since it will have to traverse all of the blocks to get to the required record, while direct access will allow us to access the required record instantly.

The direct access mechanism requires the OS to perform some additional tasks but eventually leads to much faster retrieval of records as compared to sequential access.



***Advantages of Direct/Relative Access:***

* The files can be retrieved right away with a direct access mechanism, reducing the average access time of a file.
* There is no need to traverse all of the blocks that come before the required block to access the record.

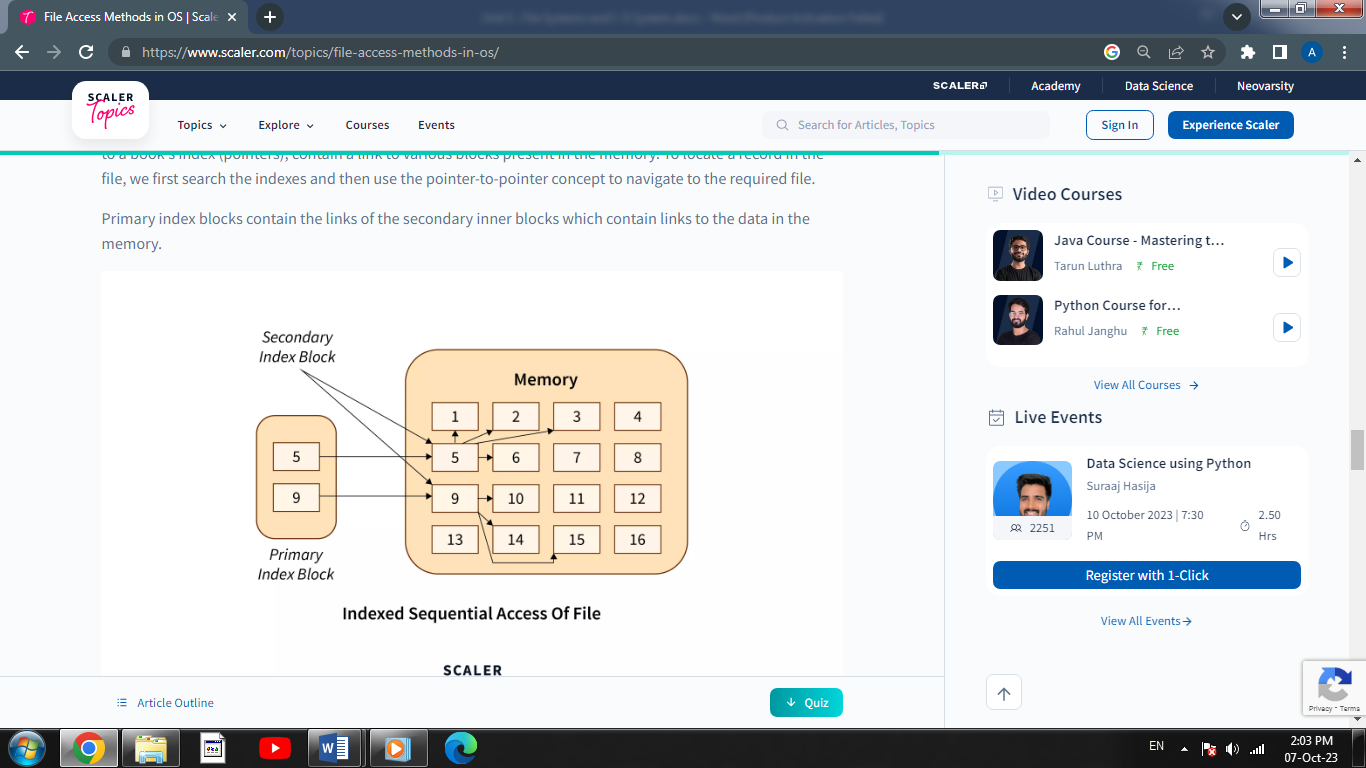
***Disadvantages of Direct/Relative Access:***

* The direct access mechanism is typically difficult to implement due to its complexity.
* Organizations can face security issues as a result of direct access as the users may access/modify the sensitive information. As a result, additional security processes must be put in place.

### **Indexed Sequential Access**

It's the other approach to accessing a file that's constructed on top of the sequential access mechanism. This method is practically similar to the pointer-to-pointer concept in which we store the address of a pointer variable containing the address of some other variable/record in another pointer variable. The indexes, similar to a book's index (pointers), contain a link to various blocks present in the memory. To locate a record in the file, we first search the indexes and then use the pointer-to-pointer concept to navigate to the required file.

Primary index blocks contain the links of the secondary inner blocks which contain links to the data in the memory.

******

***Advantages of Indexed Sequential Access:***

* If the index table is appropriately arranged, it accesses the records very quickly.
* Records can be added at any position in the file quickly.

***Disadvantages of Indexed Sequential Access:***

* When compared to other file access methods, it is costly and less efficient.
* It needs additional storage space.
* **Directory Structures in Operating System**

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.

Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that files

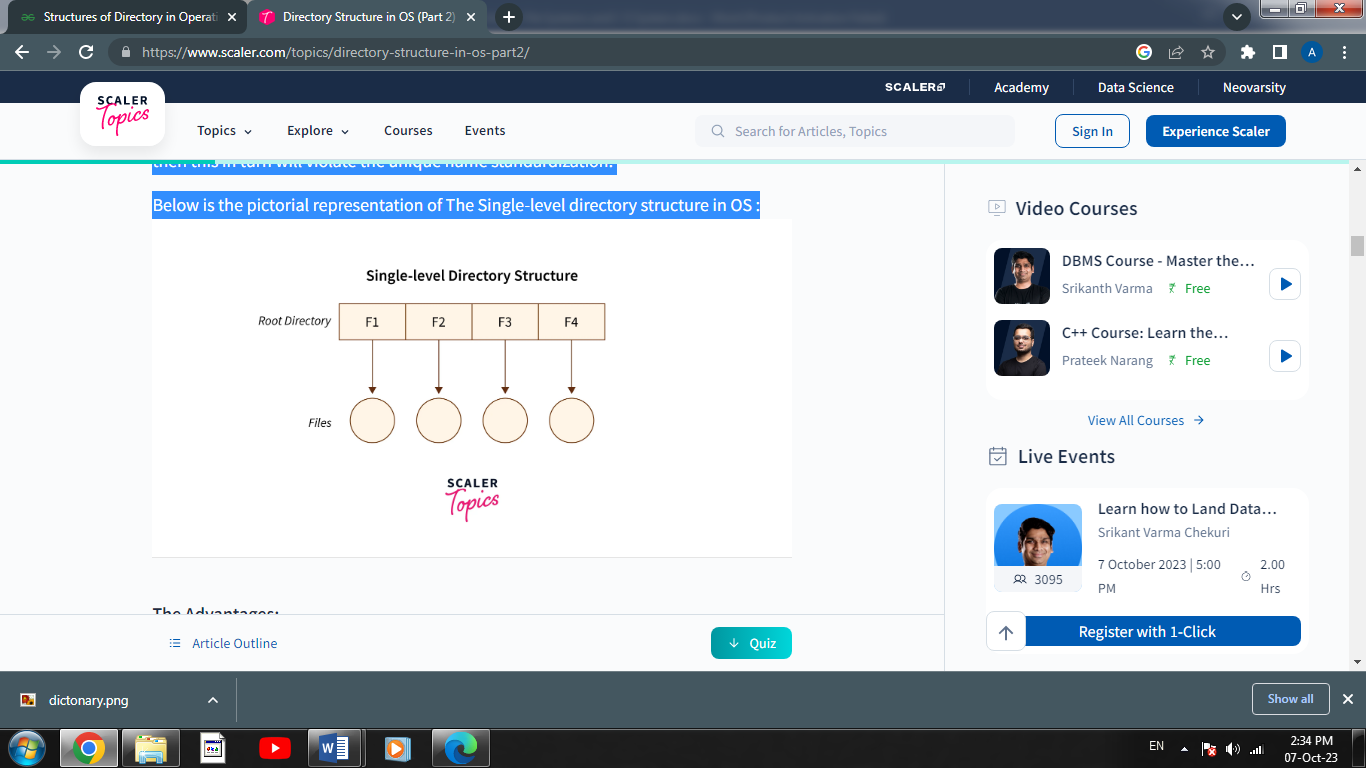
There are several logical structures of a directory, these are given below.

## The Single-Level Directory Structure:

The single-level directory structure is the simplest and easiest directory structure out of all the other directories. In this directory structure, all the folders/files are contained under the same directory which is called the root directory. As the single-level directory structure in gathering all the files under one directory or the root directory, this makes it easy to support and understand.

Now as the different files are under the same-root directory the users are not allowed to create the different sub-directories serving their requirements. This also creates a barrier with the single-level directory as when the number of files increases or more than one user logs into the system both of these need to maintain the standards of giving a unique name to it. This also means that if two users call their files 'apple', then this in turn will violate the unique name standardization.

Below is the pictorial representation of The Single-level directory structure in OS :



#### **Advantages:**

* The implementation of a single-level directory structure is simple and easier as compared to other directory structures in OS.
* If the file size is smaller, then the searching of such files with the single-level directory structure becomes simpler.
* The single-level directory structure allows the operations such as searching, creation, deletion, and updating as well.

#### **Disadvantages:**

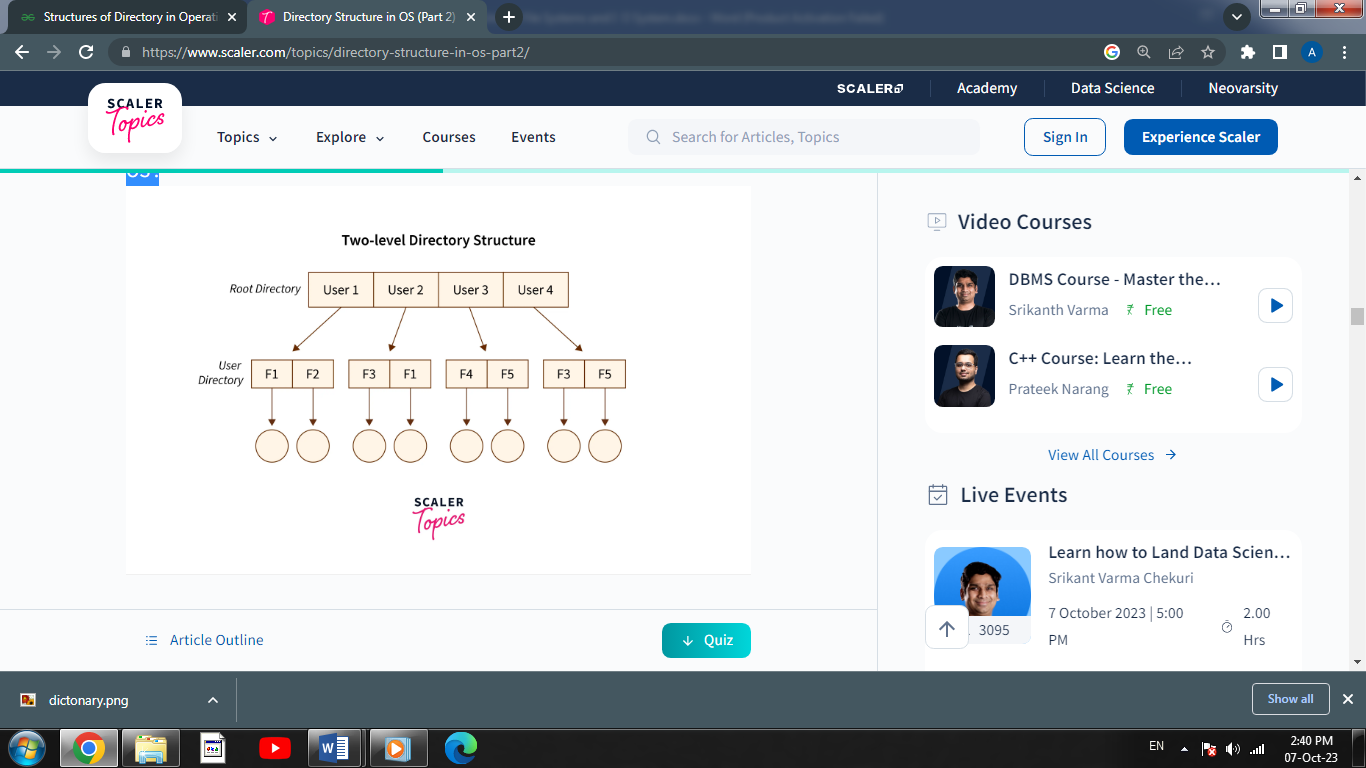
* As several users can log in at the same system for logging their files maintaining a unique name becomes difficult leading to a collision. This also means that if the file with the same name is created then the old file will get destroyed first, then the new file (having the same name ) created will be replacing it.
* If the size of the files is bigger then searching the files in one root directory of the single-level directory structure will become time taking and hence difficult.
* The single-level directory structure restricts the grouping of the same type of files together.

## The Two-Level Directory Structure :

Overcoming the drawbacks posed by the single-level directory structure,i.e, the confusion created by the same file names given by several users - The Two-level directory structure in OS came into the picture.

The two-level directory structure in OS offers a unique solution to the problem caused by single-level that is, this directory structure it gives each user the right to have their own user files directory commonly called **User File Directory(UFD)**. The User File Directory or UFDs has a similar structure as that of the single level, but each UFD lists only the files of a single user who owns that UFD. To root all the UFDs, the system’s **Master File Directory or (MFD)** searches whenever a new user id's logged into the directory structure.

This can also be defined as the two-level directory structure in OS has each of its users the right to create a directory directly inside the root directory. Here the directories created by the user are called the UFDs and to check who logged in as a user the Master File Directory or MFDs are responsible for the same.

Below is the pictorial representation of The Two-level directory structure in OS :

#### **Advantages:**

* The main advantage is there can be more than two files with same name, and would be very helpful if there are multiple users.
* A security would be there which would prevent user to access other user’s files.
* Searching of the files becomes very easy in this directory structure.

#### **Disadvantages:**

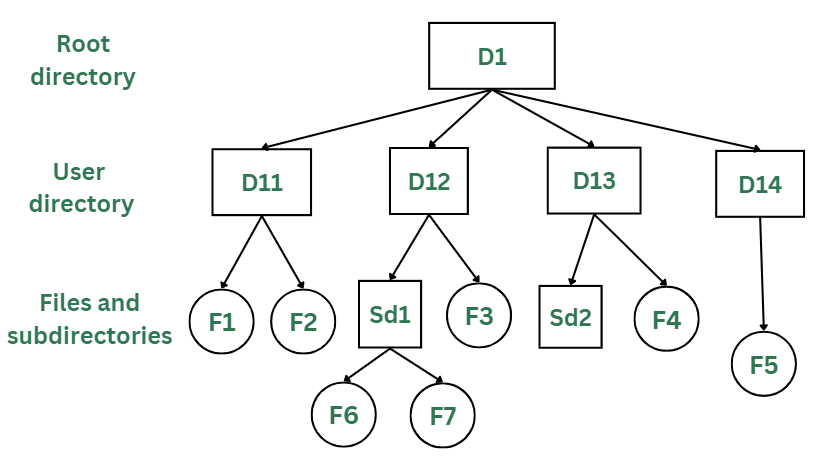
* As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
* Unlike the advantage users can create their own files, users don’t have the ability to create subdirectories.

**3. Tree Structure/ Hierarchical Structure:**

Tree directory structure of operating system is most commonly used in our **personal computers**. User can create files and subdirectories too, which was a disadvantage in the previous directory structures.

This directory structure resembles a real tree upside down, where the **root directory** is at the peak. This root contains all the directories for each user. The users can create subdirectories and even store files in their directory.

A user do not have access to the root directory data and cannot modify it. And, even in this directory the user do not have access to other user’s directories.  The structure of tree directory is given below which shows how there are files and subdirectories in each user’s directory.



#### **Advantages:**

* This directory structure allows subdirectories inside a directory.
* The searching is easier.
* File sorting of important and unimportant becomes easier.
* This directory is more scalable than the other two directory structures explained.

#### **Disadvantages:**

* As the user isn’t allowed to access other user’s directory, this prevents the file sharing among users.
* As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
* Users cannot modify the root directory data.
* If files do not fit in one, they might have to be fit into other directories.
* **Directory Operations**

1. **Creating a Directory:** This operation involves creating a new directory or folder within an existing directory. It allows users to organize their files by grouping related items together.
2. **Listing Contents:** Listing the contents of a directory displays the names of files and subdirectories contained within it. This operation helps users view the files and navigate deeper into the file system.
3. **Changing the Current Directory:** Users and programs can change their current working directory. This operation allows them to navigate to different parts of the file system easily.
4. **Renaming and Moving**: Users can rename directories and move them to different locations within the file system. Renaming is useful for giving directories more descriptive names, while moving allows for reorganization.
5. **Copying and Duplicating:** Copying directories lets users create duplicates of the directory and its contents. This operation is handy when creating backups or when you want to reuse a directory structure.
6. **Deleting a Directory:** Deleting a directory removes it and all its contents from the file system. It's essential to exercise caution when performing this operation, as it's irreversible.
7. **Monitoring Directory Changes:** Certain directory operations involve monitoring for changes within a directory, such as new files being created or existing files being modified. This is useful for real-time updates and synchronization.

* **What is Protection in Operating System?**

A mechanism that controls the access of programs, processes, or users to the resources defined by a computer system is referred to as protection. You may utilize protection as a tool for multi-programming operating systems, allowing multiple users to safely share a common logical namespace, including a directory or files.

It needs the protection of computer resources like the software, memory, processor, etc. Users should take protective measures as a helper to multiprogramming OS so that multiple users may safely use a common logical namespace like a directory or data.

Protection may be achieved by maintaining confidentiality, honesty and availability in the OS. It is critical to secure the device from unauthorized access, viruses, worms, and other malware.

## Need of Protection in Operating System

Various needs of protection in the operating system are as follows:

1. There may be security risks like unauthorized reading, writing, modification, or preventing the system from working effectively for authorized users.
2. It helps to ensure data security, process security, and program security against unauthorized user access or program access.
3. It is important to ensure no access rights' breaches, no viruses, no unauthorized access to the existing data.
4. Its purpose is to ensure that only the systems' policies access programs, resources, and data.

* **What is Operating System Security?**

## Every computer system and software design must handle all security risks and implement the necessary measures to enforce security policies. At the same time, it's critical to strike a balance because strong security measures might increase costs while also limiting the system's usability, utility, and smooth operation. As a result, system designers must assure efficient performance without compromising security.

The process of ensuring OS availability, confidentiality, integrity is known as operating system security. OS security refers to the processes or measures taken to protect the operating system from dangers, including viruses, worms, malware, and remote hacker intrusions. Operating system security comprises all preventive-control procedures that protect any system assets that could be stolen, modified, or deleted if OS security is breached.

Security refers to providing safety for computer system resources like software, CPU, memory, disks, etc. It can protect against all threats, including viruses and unauthorized access. It can be enforced by assuring the operating system's integrity, confidentiality, and availability. If an illegal user runs a computer application, the computer or data stored may be seriously damaged.

***System security may be threatened through two violations, and these are as follows:***

**1. Threat :** A program that has the potential to harm the system seriously.

**2. Attack :**A breach of security that allows unauthorized access to a resource.

There are two types of security breaches that can harm the system: malicious and accidental. Malicious threats are a type of destructive computer code or web script that is designed to cause system vulnerabilities that lead to back doors and security breaches. On the other hand, Accidental Threats are comparatively easier to protect against.

***Security may be compromised through the breaches. Some of the breaches are as follows:***

**1. Breach of integrity :**This violation has unauthorized data modification.

**2. Theft of service :** It involves the unauthorized use of resources.

**3. Breach of confidentiality :** It involves the unauthorized reading of data.

**4. Breach of availability** :It involves the unauthorized destruction of data.

**5. Denial of service** :It includes preventing legitimate use of the system. Some attacks may be accidental.

## The goal of Security System

There are several goals of system security. Some of them are as follows:

**1. Integrity :** Unauthorized users must not be allowed to access the system's objects, and users with insufficient rights should not modify the system's critical files and resources.

**2. Secrecy :** The system's objects must only be available to a small number of authorized users. The system files should not be accessible to everyone.

**3. Availability** : All system resources must be accessible to all authorized users, i.e., no single user/process should be able to consume all system resources. If such a situation arises, service denial may occur. In this case, malware may restrict system resources and preventing legitimate processes from accessing them.

* **Difference between protection and security**

| **Protection** | **Security** |
| --- | --- |
| Protection deals with who has access to the system resources. | Security gives the system access only to authorized users. |
| Protection tackles the system's internal threats. | Security tackles the system's external threats. |
| Protection addresses simple queries. | More complex queries are addressed in security. |
| It specifies which files a specific user can access or view and modify. | It defines who is permitted to access the system. |
| An authorization mechanism is used in protection. | Encryption and certification (authentication) mechanisms are implemented. |
| Protection provides a mechanism for controlling access to processes, programs, and user resources. | While security provides a mechanism to safeguard the system resources and the user resources from all external users. |

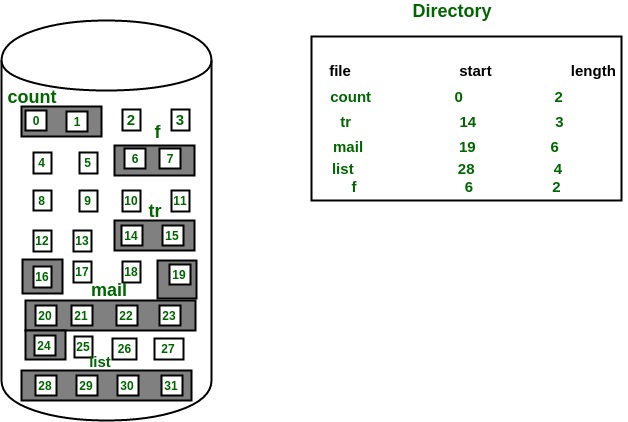
* **File Allocation Methods**

1. **Contiguous Allocation**
2. **Linked Allocation**
3. **Indexed Allocation**
4. **Contiguous Allocation**

In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.  
The directory entry for a file with contiguous allocation contains

* ***Address of starting block***
* ***Length of the allocated portion.***

The file ‘mail’ in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



***Advantages:***

* Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
* This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

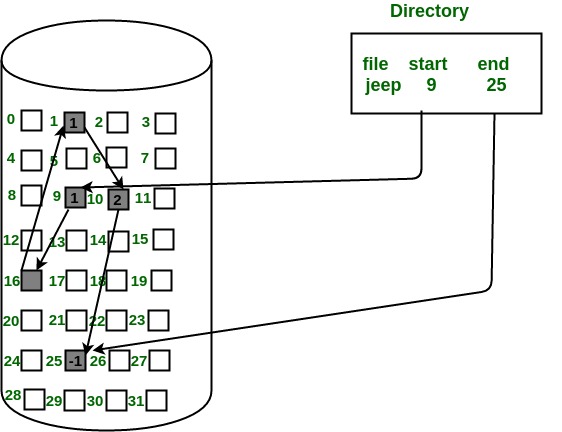
***Disadvantages:***

* This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
* Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

1. **Linked List Allocation**

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.  
The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file ‘jeep’ in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



***Advantages:***

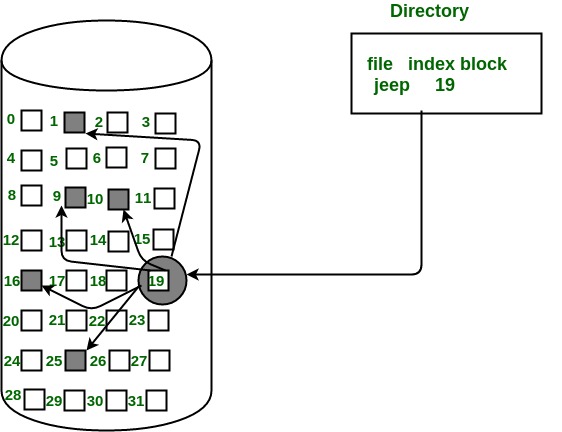
* This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
* This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

***Disadvantages:***

* Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
* It does not support random or direct access. We cannot directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access ) from the starting block of the file via block pointers.
* Pointers required in the linked allocation incur some extra overhead.

1. **Indexed Allocation**

In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:



***Advantages:***

* This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
* It overcomes the problem of external fragmentation.

***Disadvantages:***

* The pointer overhead for indexed allocation is greater than linked allocation.
* For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.
* **Input/output System**

The three main jobs of a computer are Input, Output, and Processing. In most of the cases, the most important job is Input / Output, and the processing is simply incidental. For an example, when we browse a web page or edit any file, our immediate attention is to read or enter some information, not for computing an answer. The fundamental role of the operating system in computer Input / Output is to manage and organize I/O operations and all I/O devices.

The various devices that are connected to the computer need to be controlled and it is a key concern of operating-system designers. This is as I/O devices vary so widely in their functionality and speed (for example a mouse, a hard disk and a CD-ROM), varied methods are required for controlling them. These methods form the I/O sub-system of the kernel of OS that separates the rest of the kernel from the complications of managing I/O devices.

* **Principles of I/O hardware**

Different people look at I/O hardware in different ways. Electrical engineers look at it in terms of chips, wires, power supplies, motors, and all the other physical components that make up the hardware. Programmers look at the interface presented to the software the commands the hardware accepts, the functions it carries out, and the errors that can be reported back.

In this book we are concerned with programming I/O devices, not designing, building, or maintaining them, so our interest will be restricted to how the hardware is programmed, not how it works inside. Nevertheless, the programming of many I/O devices is often intimately connected with their internal operation. In the next three subsections we will provide a little general background on I/O hardware as it relates to programming.

* **Input and Output Devices**

**Input Devices:**

[Input devices](https://www.geeksforgeeks.org/what-are-input-devices/) are the devices that are used to send signals to the computer for performing tasks. The receiver at the end is the CPU (Central Processing Unit), which has work to send signals to the output devices. Some of the classifications of Input devices are:

* Keyboard Devices
* Pointing Devices
* Composite Devices
* Game Controller
* Visual Devices
* Audio Input Devices

***Some of the input devices are described below.***

**Keyboard:**

The keyboard is the most frequent and widely used input device for entering data into a computer. Although there are some additional keys for performing other operations, the keyboard layout is similar to that of a typical typewriter.  
Generally, keyboards come in two sizes: 84 keys or 101/102 keys but currently keyboards with 104 keys or 108 keys are also available for Windows and the Internet.

**Mouse :**

The most common pointing device is the mouse. The mouse is used to move a little cursor across the screen while clicking and dragging. The cursor will stop if you let go of the mouse. The computer is dependent on you to move the mouse; it won’t move by itself. As a result, it’s an input device.  
A mouse is an input device that lets you move the mouse on a flat surface to control the coordinates and movement of the on-screen cursor/pointer.  
The left mouse button can be used to select or move items, while the right mouse button when clicked displays extra menus.

**Joystick :**

The joystick’s function is comparable to that of a mouse. It is primarily used in CAD (Computer-Aided Design) and playing video games on the computer.

**Scanner**

A scanner is an input device that functions similarly to a photocopier. It’s employed when there’s information on paper that needs to be transferred to the computer’s hard disc for subsequent manipulation. The scanner collects images from the source and converts them to a digital format that may be saved on a disc. Before they are printed, these images can be modified.

**Bar Code Reader :**

A bar code reader is a device that reads data that is bar-coded (data that is represented by light and dark lines).Bar-coded data is commonly used to mark things, number books, and so on. It could be a handheld scanner or part of a stationary scanner. A bar code reader scans a bar code image, converts it to an alphanumeric value, and then sends it to the computer to which it is connected.

**Web Camera**

Because a web camera records a video image of the scene in front of it, a webcam is an input device. It is either built inside the computer (for example, a laptop) or attached through a USB connection. A webcam is a computer-connected tiny digital video camera. It’s also known as a web camera because it can take images and record video. These cameras come with software that must be installed on the computer in order to broadcast video in real-time over the Internet. It can shoot images and HD videos, however, the video quality isn’t as good as other cameras (In Mobiles or other devices or normal cameras).

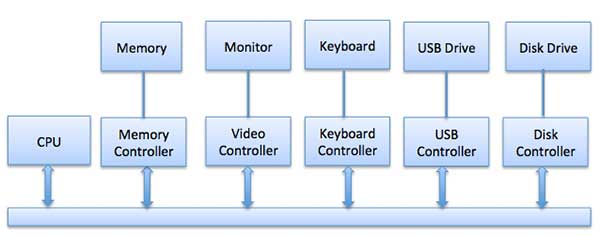
* **Device Controller**

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.



# 

* **What is a DMA (Direct Memory Access)**

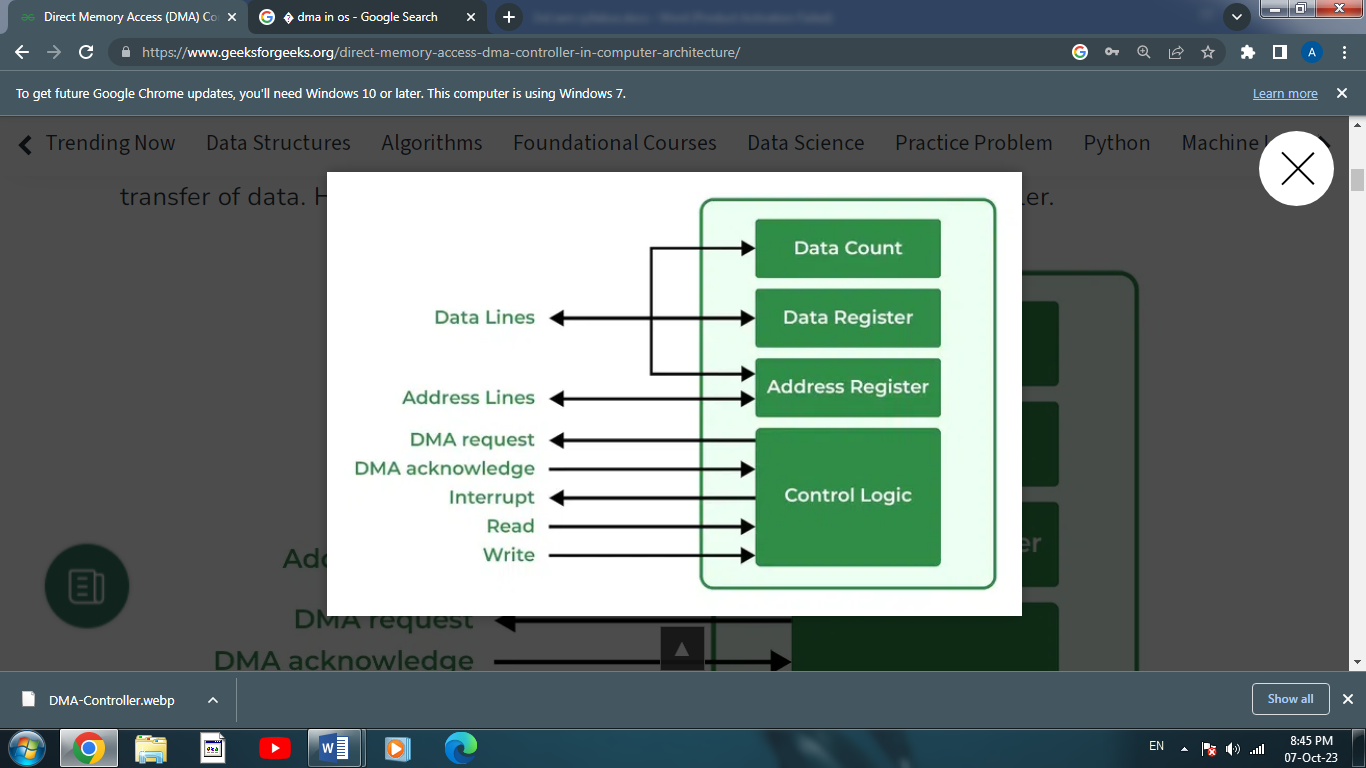
[Direct Memory Access](https://www.geeksforgeeks.org/direct-memory-access-with-dma-controller-8257-8237/) uses hardware for accessing the memory, that hardware is called a DMA Controller. It has the work of transferring the data between Input Output devices and main memory with very less interaction with the processor. The direct Memory Access Controller is a control unit, which has the work of transferring data.

DMA Controller is a hardware device that allows I/O devices to directly access memory with less participation of the processor. DMA controller needs the same old circuits of an interface to communicate with the CPU and Input/Output devices.

## DMA Controller Diagram in Computer Architecture

DMA Controller is a type of control unit that works as an interface for the data bus and the I/O Devices. As mentioned, DMA Controller has the work of transferring the data without the intervention of the processors, processors can control the data transfer.

DMA Controller also contains an address unit, which generates the address and selects an I/O device for the transfer of data. Here we are showing the block diagram of the DMA Controller.



## Types of Direct Memory Access (DMA)

There are four popular types of DMA.

1. **Single-Ended DMA:** Single-Ended DMA Controllers operate by reading and writing from a single memory address. They are the simplest DMA.
2. **Dual-Ended DMA:**Dual-Ended DMA controllers can read and write from two memory addresses. Dual-ended DMA is more advanced than single-ended DMA.
3. **Arbitrated-Ended DMA:**Arbitrated-Ended DMA works by reading and writing to several memory addresses. It is more advanced than Dual-Ended DMA.
4. **Interleaved DMA:**Interleaved DMA are those DMA that read from one memory address and write from another memory address.

## *Advantages of DMA Controller*

* Data Memory Access speeds up memory operations and data transfer.
* CPU is not involved while transferring data.
* DMA requires very few clock cycles while transferring data.
* DMA distributes workload very appropriately.
* DMA helps the CPU in decreasing its load.

## *Disadvantages of DMA Controller*

* Direct Memory Access is a costly operation because of additional operations.
* DMA suffers from [Cache-Coherence Problems](https://www.geeksforgeeks.org/cache-coherence/).
* DMA Controller increases the overall cost of the system.
* DMA Controller increases the complexity of the software.
* **Principles of I/O software**

**Goals of I/O Software:**

The primary goal of I/O software in an operating system is to provide a layer of abstraction and management for input and output operations. It aims to hide the low-level hardware details of different devices from application programs, making it easier for developers to interact with these devices.

I/O software also strives to enhance system performance and efficiency by optimizing data transfer between the CPU and various I/O devices. It should ensure data integrity and reliability during I/O operations, handling errors and failures gracefully.

### **Device Drivers**

Software modules that can be plugged into an OS in order to handle any device are known as device drivers. They help the OS handle I/O devices. Device drivers have device-dependent codes containing device-specific register reads/writes that help implements a standard interface. They are written by a device’s manufacturer and are stored on a CD-ROM and sent with the device.

***The job of a device driver is to:***

* Accept requests from the device-independent software
* Interact with the device controller in order to take/give inputs and outputs
* Perform error handling as and when required
* Make sure that the execution of requests is successful

If the device driver is idle when a request arrives, it proceeds to carry out the request. Otherwise, it places the new request in the queue of pending requests.

### **Interrupt handlers**

An interrupt handler or interrupt service routine (ISR), is a part of software. It is more like a callback function in an OS (device driver) that starts execution after receiving an interrupt. Thus, interrupt is like a trigger for interrupt handlers.

The interrupt takes many measures to handle an interrupt like updating data structures and waking up the process waiting for the interrupt. The interrupt mechanism accepts an address, an offset stored in the interrupt vector table.

* An interrupt handler is a crucial component of I/O software. It is a piece of code that responds to hardware interrupts generated by I/O devices.
* When an I/O device needs attention (e.g., data is ready for input, or an operation is completed), it sends an interrupt signal to the CPU.
* The interrupt handler is responsible for temporarily suspending the currently executing program, saving its state, and then handling the interrupt.
* Depending on the type of interrupt (e.g., hardware or software interrupt), the handler may perform different tasks, such as initiating or completing I/O operations, handling errors, or updating device status.
* **Mass Storage Structure**

Systems designed to store enormous volumes of data are referred to as mass storage devices. Massive storage devices are sometimes used interchangeably with peripheral storage, which is the management of bigger volumes of data that are larger than the native storage capability of a computer or device.

The basic idea of Mass Storage is to create a Data Backup or Data Recovery System.

Along with computer systems, definitions of mass storage technologies and tactics have changed. The earliest and most basic mass storage techniques date back to the era of main frame supercomputers, according to experts. Punch cards, Hollerith cards, and other relatively similar manual storage medium are examples of this Mass Storage Media these days. Today, mass storage may include several kinds of hard disks or solid-state storage devices, as well as tape drives and other physical data storage devices.

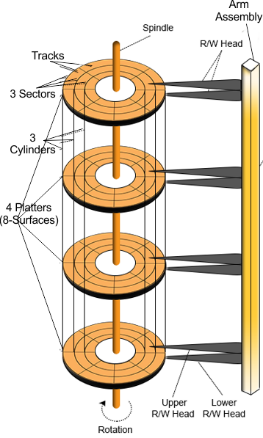
The concepts of data backup and data recovery are frequently linked to mass storage media. The biggest Business Companies will make plans for recording, storing, and backing up all accessible data, which calls for a lot more mass storage media than what factory-direct gear can provide. This suggests a method for handling continuous mass storage that uses tape or other media. Other kinds of mass storage could function well as a data storage plan for a big network or a bunch of mobile distant devices. To backup data on a portable tablet that doesn't have a lot of internal capacity, for instance, mass storage for a tablet might require the usage of flash or Universal Serial Bus media (USB Media).

The Mass Storage Structure Devices are:

1. Magnetic Disks
2. Solid State Disks
3. Magnetic Tapes

* **Disk Structure In OS**

Here we are taking only hard disk structure. Hard disk is secondary storage which stores the large amount of data. The Hard disk drive contains a dozens of disks.  These disks are also known as platters. These platters are mount over the spindle which rotates in any direction i.e. clockwise or anti-clockwise. Lets look at hard disk structure in [OS](https://cstaleem.com/operating-system).



## Platter

* The Platter is made of aluminum or iron oxide.
* Platter diameter range is 1.8 inches to 5.25 inches
* Each platter contains 2 surfaces and 2 Read/Write Head. One Read/Write head requires for one surface of the platter. So, other R/W head use for other surface to store the information’s.
* Every platter holds the same no. of tracks.
* Multiple platters increase the storage capacity.

A single platter descriptive diagram is given below



## R/W Head

* R/W Heads moves forth and back over the platter surfaces to Read or Write the data on sectors.
* Read/Write heads does not touch to platters surface. The data written over the platter surface is done through magnetic field. If R/W Head touches over the surface of platter then bad sectors may creates. Had disk may damage due to these bad sectors.

## Tracks

* Circular areas of disk are known as tracks.
* There may be more than a 1000 tracks on a 3.5 inch hard disk and sector size. Track Numbering start with zero at outermost track.

## Sectors

Tracks are further divided into number of small units; these small units are known as sectors. Sectors are the smallest physical storage units on disk. Size of each sector is almost always 512 Bytes. Sector Numbering start with 1 at outermost tracks.

## Cylinder

All Corresponding tracks with same radius of all platters in the Hard disk are known as cylinders.  In simple words we say “Each tack of all platters with same radius is called a cylinder”

So, Number of tracks in platter is always equal to number of cylinders. For example, a hard disk, where each platter contains 600 tracks then the number of cylinders will also be 600 in the hard disk.

Cylinder Numbering start with zero at outermost cylinder

* **Disk Scheduling**

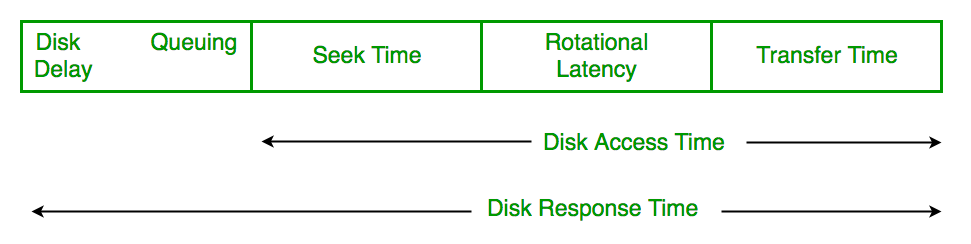
**Disk scheduling**is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O Scheduling.

## *Importance of Disk Scheduling in Operating System*

* Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
* Two or more requests may be far from each other so this can result in greater disk arm movement.
* Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

## *Key Terms Associated with Disk Scheduling*

* **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written. So the disk scheduling algorithm that gives a minimum average seek time is better.
* **Rotational Latency:** Rotational Latency is the time taken by the desired sector of the disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
* **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and the number of bytes to be transferred.



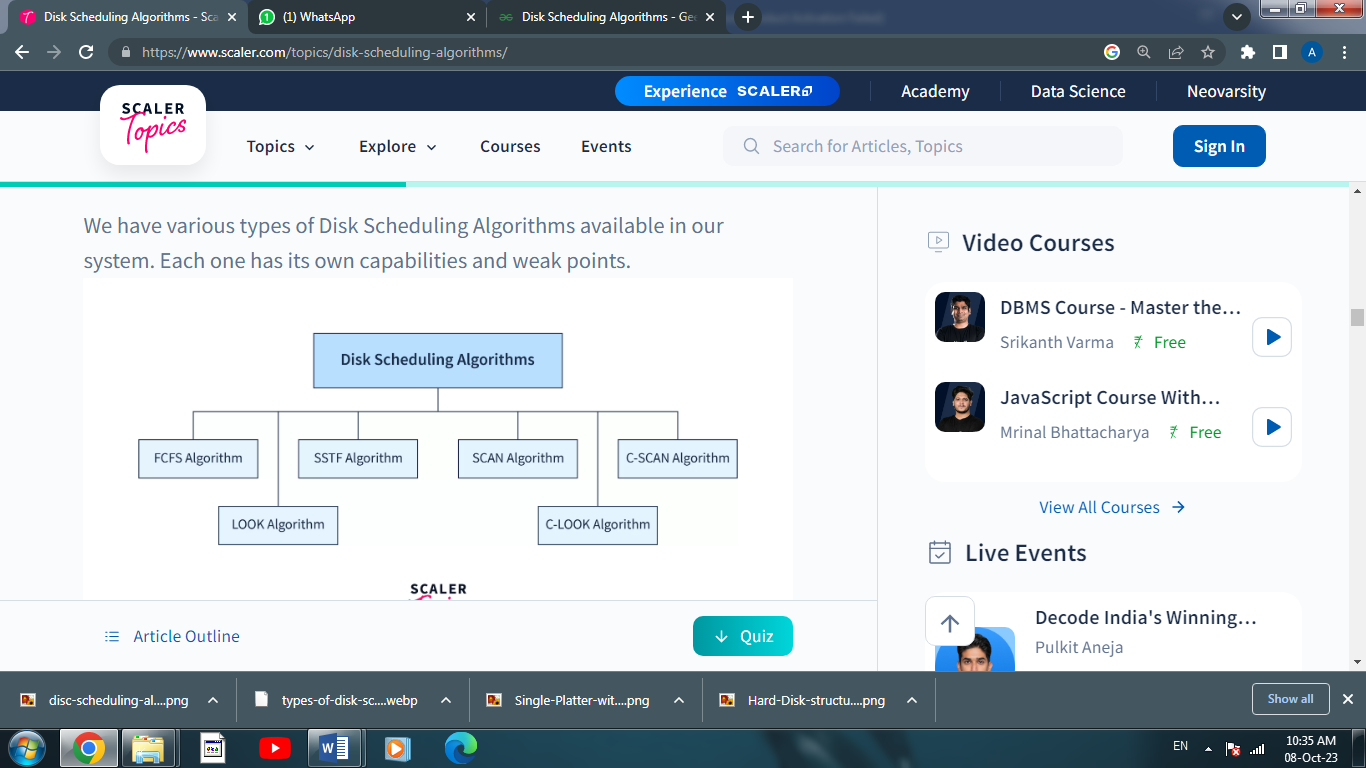
* **Disk Access Time:**

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

Total Seek Time = Total head Movement \* Seek Time

**Disk Response Time:**Response Time is the average time spent by a request waiting to perform its I/O operation. The average*Response time*is the response time of all requests. *Variance Response Time*is the measure of how individual requests are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

* **Disk Scheduling Algorithms**



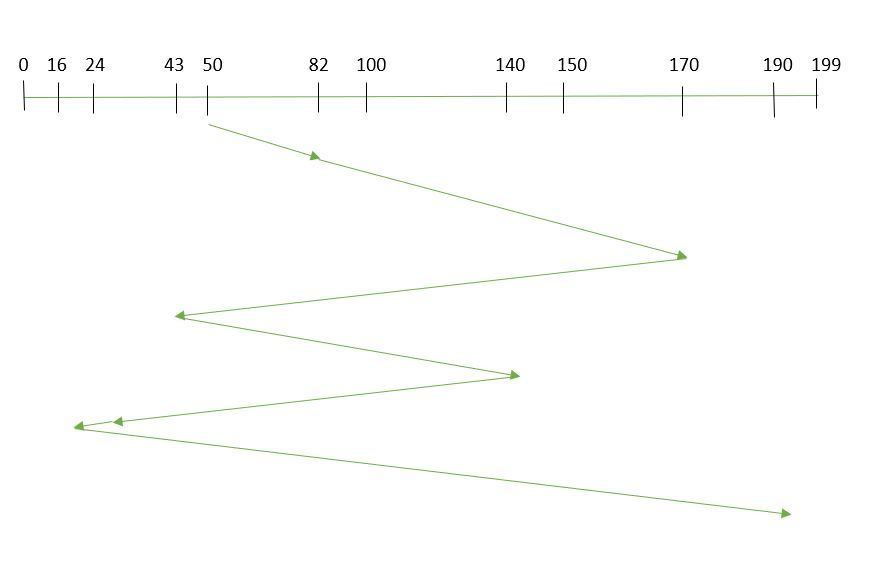
1. **FCFS (First Come First Serve)**

[FCFS](https://www.geeksforgeeks.org/fcfs-disk-scheduling-algorithms/) is the simplest of all Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.

It stands for 'first-come-first-serve'. As the name suggests, the request that comes first will be processed first and so on. The requests coming to the disk are arranged in a proper sequence as they arrive. Since every request is processed in this algorithm, so there is no chance of 'starvation'.

**Example:**

Suppose the order of request is- **(82,170,43,140,24,16,190)**  
And current position of Read/Write **head is: 50**



So**, Total overhead movement**  (total distance covered by the disk arm) = (82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) **=642**

***Advantages of FCFS***

* Here are some of the advantages of First Come First Serve.
* Every request gets a fair chance
* No indefinite postponement

***Disadvantages of FCFS***

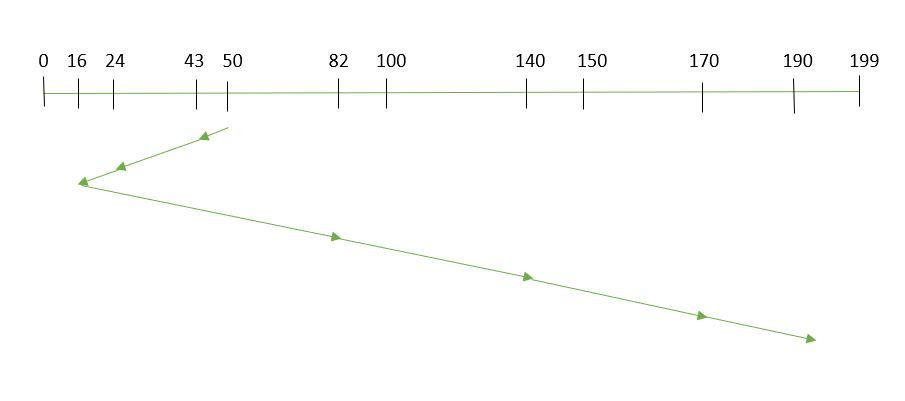
* Here are some of the disadvantages of First Come First Serve.
* Does not try to optimize seek time
* May not provide the best possible service

1. **SSTF (Shortest Seek Time First)**

In [SSTF (Shortest Seek Time First)](https://www.geeksforgeeks.org/program-for-sstf-disk-scheduling-algorithm/), requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of the system. Let us understand this with the help of an example.

**Example:**

Suppose the order of request is- (**82,170,43,140,24,16,190**).  
And current position of Read/Write **head is: 50**



**Total overhead movement** (total distance covered by the disk arm) =

(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) =**208**

***Advantages of Shortest Seek Time First***

* Here are some of the advantages of Shortest Seek Time First.
* The average Response Time decreases
* Throughput increases

***Disadvantages of Shortest Seek Time First***

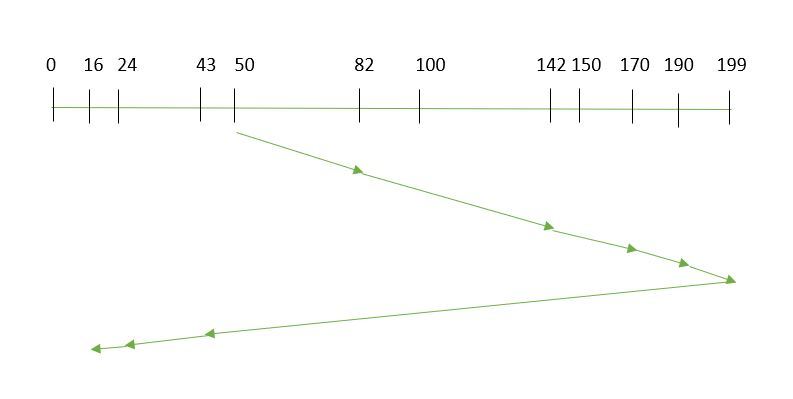
* Here are some of the disadvantages of Shortest Seek Time First.
* Overhead to calculate seek time in advance
* Can cause Starvation for a request if it has a higher seek time as compared to incoming requests
* The high variance of response time as SSTF favors only some requests

1. **SCAN**

In the [SCAN algorithm](https://www.geeksforgeeks.org/scan-elevator-disk-scheduling-algorithms/) the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and is hence also known as an **elevator algorithm.**As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

**Example:**

Suppose the requests to be addressed are**-82,170,43,140,24,16,190**. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**

****

Therefore, the **Total overhead movement** (total distance covered by the disk arm)  is calculated as

= (199-50) + (199-16) = **332**

***Advantages of SCAN Algorithm***

* Here are some of the advantages of the SCAN Algorithm.
* High throughput
* Low variance of response time
* Average response time

***Disadvantages of SCAN Algorithm***

* Here are some of the disadvantages of the SCAN Algorithm.
* Long waiting time for requests for locations just visited by disk arm

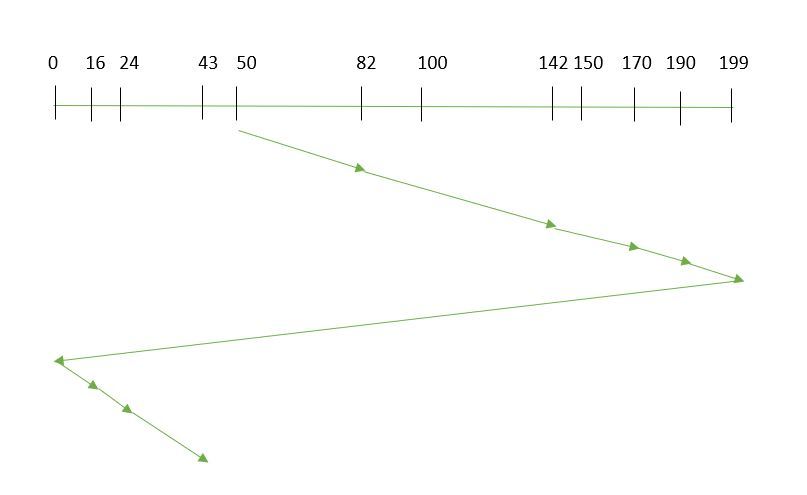
1. **C-SCAN**

In the [SCAN algorithm](https://www.geeksforgeeks.org/c-scan-disk-scheduling-algorithm/), the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in the *CSCAN*algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to the SCAN algorithm hence it is known as C-SCAN (Circular SCAN).

**Example:**

Suppose the requests to be addressed are-**82, 170,43,140,24,16,190**. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**



So, the **Total overhead movement** (total distance covered by the disk arm) is calculated as:

=(199-50) + (199-0) + (43-0) **= 391**

***Advantages of C-SCAN Algorithm***

Here are some of the advantages of C-SCAN.

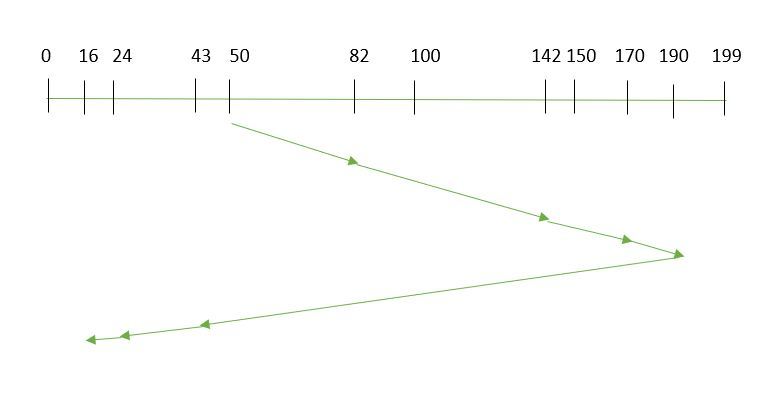
Provides more uniform wait time compared to SCAN.

1. **LOOK**

[LOOK Algorithm](https://www.geeksforgeeks.org/look-disk-scheduling-algorithm/) is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

**Example:**

Suppose the requests to be addressed are-**82,170,43,140,24,16,190**. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**



So, the **Total overhead movement**  (total distance covered by the disk arm) is calculated as:

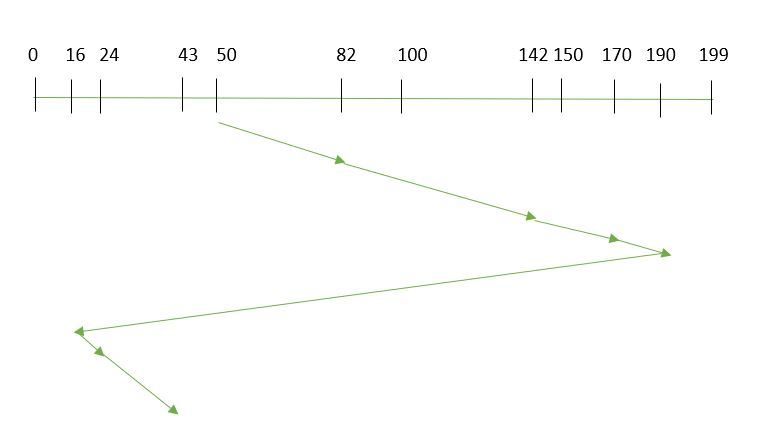
= (190-50) + (190-16) = **314**

1. **C-LOOK**

As LOOK is similar to the SCAN algorithm, in a similar way, [C-LOOK](https://www.geeksforgeeks.org/c-look-disk-scheduling-algorithm/) is similar to the CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end’s last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

**Example:**

Suppose the requests to be addressed are-**82, 170,43,140,24,16,190**. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”**



So, the **Total overhead movement** (total distance covered by the disk arm) is calculated as

= (190-50) + (190-16) + (43-16) **= 341**